

PATENT APPLICATION
ATTORNEY DOCKET NO. SUN-P6432-SPL

5

10 **METHOD AND APPARATUS FOR SOLVING
AN INEQUALITY CONSTRAINED GLOBAL
OPTIMIZATION PROBLEM**

Inventors: G. William Walster and Eldon R. Hansen

15

Related Application

20 The subject matter of this application is related to the subject matter in a
co-pending non-provisional application by the same inventors as the instant
application entitled, "Applying Term Consistency to an Inequality Constrained
Interval Global Optimization Problem," having serial number TO BE
ASSIGNED, and filing date 13 December 2001 (Attorney Docket No. SUN-
P6446-SPL).

25

BACKGROUND

Field of the Invention

30 The present invention relates to performing arithmetic operations on
interval operands within a computer system. More specifically, the present

invention relates to a method and an apparatus for using a computer system to solve a global optimization problem including inequality constraints with interval arithmetic.

5 **Related Art**

Rapid advances in computing technology make it possible to perform trillions of computational operations each second. This tremendous computational speed makes it practical to perform computationally intensive tasks as diverse as predicting the weather and optimizing the design of an aircraft
10 engine. Such computational tasks are typically performed using machine-representable floating-point numbers to approximate values of real numbers. (For example, see the Institute of Electrical and Electronics Engineers (IEEE) standard 754 for binary floating-point numbers.)

In spite of their limitations, floating-point numbers are generally used to
15 perform most computational tasks.

One limitation is that machine-representable floating-point numbers have a fixed-size word length, which limits their accuracy. Note that a floating-point number is typically encoded using a 32, 64 or 128-bit binary number, which means that there are only 2^{32} , 2^{64} or 2^{128} possible symbols that can be used to
20 specify a floating-point number. Hence, most real number values can only be approximated with a corresponding floating-point number. This creates estimation errors that can be magnified through even a few computations, thereby adversely affecting the accuracy of a computation.

A related limitation is that floating-point numbers contain no information
25 about their accuracy. Most measured data values include some amount of error that arises from the measurement process itself. This error can often be quantified as an accuracy parameter, which can subsequently be used to determine the

accuracy of a computation. However, floating-point numbers are not designed to keep track of accuracy information, whether from input data measurement errors or machine rounding errors. Hence, it is not possible to determine the accuracy of a computation by merely examining the floating-point number that results from
5 the computation.

Interval arithmetic has been developed to solve the above-described problems. Interval arithmetic represents numbers as intervals specified by a first (left) endpoint and a second (right) endpoint. For example, the interval $[a, b]$, where $a < b$, is a closed, bounded subset of the real numbers, R , which includes a
10 and b as well as all real numbers between a and b . Arithmetic operations on interval operands (interval arithmetic) are defined so that interval results always contain the entire set of possible values. The result is a mathematical system for rigorously bounding numerical errors from all sources, including measurement data errors, machine rounding errors and their interactions. (Note that the first
15 endpoint normally contains the “infimum”, which is the largest number that is less than or equal to each of a given set of real numbers. Similarly, the second endpoint normally contains the “supremum”, which is the smallest number that is greater than or equal to each of the given set of real numbers.)

One commonly performed computational operation is to perform
20 inequality constrained global optimization to find a global minimum of a nonlinear objective function $f(\mathbf{x})$, subject to nonlinear inequality constraints of the form $p_i(\mathbf{x}) \leq 0$ ($i=1, \dots, m$). This can be accomplished using any members of a set of criteria to delete boxes, or parts of boxes that either fail to satisfy one or more inequality constraints, or cannot contain the global minimum f^* given the
25 inequality constraints are all satisfied. The set of criteria includes:

(1) the f_bar -criterion, wherein if f_bar is the smallest upper bound so far computed on f within the feasible region defined by the inequality constraints,

then any point \mathbf{x} for which $f(\mathbf{x}) > f_bar$ can be deleted. Similarly, any box \mathbf{X} can be deleted if $\inf(f(\mathbf{X})) > f_bar$;

(2) the monotonicity criterion, wherein if $\mathbf{g}(\mathbf{x})$ is the gradient of f evaluated at a feasible point \mathbf{x} for which all $p_i(\mathbf{x}) < 0$ ($i=1, \dots, m$), then any such
5 feasible point \mathbf{x} for which $\mathbf{g}(\mathbf{x}) \neq \mathbf{0}$ can be deleted. Similarly, any feasible box \mathbf{X} can be deleted if $0 \notin \mathbf{g}(\mathbf{X})$;

(3) the convexity criterion, wherein if $H_{ii}(\mathbf{x})$ is the i -th diagonal element of the Hessian of f , then any feasible point \mathbf{x} for all which all $p_i(\mathbf{x}) < 0$ ($i=1, \dots, m$) and $H_{ii}(\mathbf{x}) < 0$ (for $i=1, \dots, n$) can be deleted. Similarly, any box \mathbf{X} in
10 the interior of the feasible region can be deleted if $H_{ii}(\mathbf{X}) < 0$ (for $i=1, \dots, n$); and

(4) the stationary point criterion, wherein points \mathbf{x} are deleted using the interval Newton technique to solve the John conditions. (The John conditions are described in "Global Optimization Using Interval Analysis" by Eldon R. Hansen, Marcel Dekker, Inc., 1992.)

15 All of these criteria work best "in the small" when the objective function f is approximately linear or quadratic and "active" constraints are approximately linear. An active constraint is one that is zero at a solution point. For large intervals containing multiple stationary points the above criteria might not succeed in deleting much of a given box. In this case the box is split into two or
20 more sub-boxes, which are then processed independently. By this mechanism *all* the inequality constrained global minima of a nonlinear objective function can be found.

One problem is applying this procedure to large n -dimensional interval vectors (or boxes) that contain multiple local minima. In this case, the process of
25 splitting in n -dimensions can lead to exponential growth in the number of boxes to process.

It is well known that this problem (and even the problem of computing “sharp” bounds on the range of a function of n -variables over an n -dimensional box) is an “NP-hard” problem. In general, NP-hard problems require an exponentially increasing amount of work to solve as n , the number of independent variables, increases.

Because NP-hardness is a worst-case property and because many practical engineering and scientific problems have relatively simple structure, one problem is to use this simple structure of real problems to improve the efficiency of interval inequality constrained global optimization algorithms.

Hence, what is needed is a method and an apparatus for using the structure of a nonlinear objective function to improve the efficiency of interval inequality constrained global optimization software. To this end, what is needed is a method and apparatus that efficiently deletes “large” boxes or parts of large boxes that the above criteria can only split.

SUMMARY

The present invention combines various methods to speed up the process of bounding all the inequality constrained global minima of a nonlinear function. The combined method uses the structure of the objective function and inequality constraints to efficiently delete parts or all of large boxes that would otherwise have to be split. One embodiment of the present invention provides a system that solves a global inequality constrained optimization problem specified by a function f and a set of inequality constraints $p_i(\mathbf{x}) \leq 0$ ($i=1, \dots, m$), wherein f and p_i are scalar functions of a vector $\mathbf{x} = (x_1, x_2, x_3, \dots, x_n)$. During operation, the system receives a representation of the function f and the set of inequality constraints, and stores the representation in a memory within the computer system. Next, the system performs an interval inequality constrained global optimization process to

compute guaranteed bounds on a globally minimum value of the function $f(\mathbf{x})$ subject to the set of inequality constraints. During this process, the system applies term consistency to a set of relations associated with the global inequality constrained optimization problem over a subbox \mathbf{X} , and excludes any portion of the subbox \mathbf{X} that violates any set of these relations. The system also applies box consistency to the set of relations associated with the global inequality constrained optimization problem over the subbox \mathbf{X} , and excludes any portion of the subbox \mathbf{X} that violates any of the relations. The system also performs an interval Newton step for the global inequality constrained optimization problem over the subbox \mathbf{X} to produce a resulting subbox \mathbf{Y} , wherein the point of expansion of the interval Newton step is a point \mathbf{x} .

For any function of n -variables $f(\mathbf{x})$ there are different ways to analytically express a component x_j of the vector \mathbf{x} . For example, one can write $f(\mathbf{x}) = g(x_j) - h(\mathbf{x})$, where $g(x_j)$ is a term in f for which it is possible to solve $g(x_j) = y$ for an element of \mathbf{x} , x_j , using $g^{-1}(y)$. For each of these rearrangements, if a given interval box \mathbf{X} is used as an argument of h , then the new interval X_j^+ for the j -th component of \mathbf{X} , is guaranteed to be at least as narrow as the original, X_j .

$$X_j^+ = X_j \cap X'_j \text{ where } X'_j = g^{-1}(h(\mathbf{X})).$$

This process is then be iterated using different terms g of the function f . After reducing any element X_j of the box \mathbf{X} to X_j^+ , the reduced value can be used in \mathbf{X} thereafter to speed up the reduction process using other component functions if f is a component of the vector function \mathbf{f} .

Hereafter, the notation $g(x_j)$ for a term of the function $f(\mathbf{x})$ implicitly represents any term of any component function. This eliminates the need for additional subscripts that do not add clarity to the exposition.

In one embodiment of the present invention, while applying term consistency, the system symbolically manipulates an equation to solve for a term, $g(x'_j)$, thereby producing a modified equation $g(x'_j) = h(\mathbf{x})$, wherein the term $g(x'_j)$ can be analytically inverted to produce an inverse function $g^{-1}(y)$. Next, the system substitutes the subbox \mathbf{X} into the modified equation to produce the equation $g(X'_j) = h(\mathbf{X})$. The system then solves for $X'_j = g^{-1}(h(\mathbf{X}))$; and intersects X'_j with the j -th element of the subbox \mathbf{X} to produce a new subbox \mathbf{X}^+ . This new subbox \mathbf{X}^+ contains all solutions of the equation within the subbox \mathbf{X} . Furthermore the size of the new subbox \mathbf{X}^+ is less than or equal to the size of the subbox \mathbf{X} .

In a variation on this embodiment, applying term consistency to the set of relations involves applying term consistency to the set of inequality constraints $p_i(\mathbf{x}) \leq 0$ ($i=1, \dots, m$) over the subbox \mathbf{X} .

In a variation on this embodiment, applying box consistency to the set of relations involves applying box consistency to the set of inequality constraints $p_i(\mathbf{x}) \leq 0$ ($i=1, \dots, m$) over the subbox \mathbf{X} .

In a variation on this embodiment, during the interval inequality constrained global optimization process, the system keeps track of a least upper bound f_bar of the function $f(\mathbf{x})$ at a feasible point \mathbf{x} , and removes from consideration any subbox \mathbf{X} for which $f(\mathbf{X}) > f_bar$. In this variation, the system applies term consistency to the f_bar inequality $f(\mathbf{x}) \leq f_bar$ over the subbox \mathbf{X} .

In a further variation, the system applies box consistency to the f_bar inequality $f(\mathbf{x}) \leq f_bar$ over the subbox \mathbf{X} .

In a variation on this embodiment, if the subbox \mathbf{X} is strictly feasible ($p_i(\mathbf{X}) < 0$ for all $i=1, \dots, n$), the system determines a gradient $\mathbf{g}(\mathbf{x})$ of the function $f(\mathbf{x})$, wherein $\mathbf{g}(\mathbf{x})$ includes components $g_i(\mathbf{x})$ ($i=1, \dots, n$). Next, the system removes from consideration any subbox for which $\mathbf{g}(\mathbf{x})$ is bounded away from

zero, indicating that the subbox does not include an extremum of $f(\mathbf{x})$. In this variation, the system applies term consistency to each component $g_i(\mathbf{x})=0$ ($i=1, \dots, n$) of $\mathbf{g}(\mathbf{x})=\mathbf{0}$ over the subbox \mathbf{X} .

In a further variation, the system applies box consistency to each
5 component $g_i(\mathbf{x})=0$ ($i=1, \dots, n$) of $\mathbf{g}(\mathbf{x})=\mathbf{0}$ over the subbox \mathbf{X} .

In a variation on this embodiment, if the subbox \mathbf{X} is strictly feasible ($p_i(\mathbf{X}) < 0$ for all $i=1, \dots, n$), the system determines diagonal elements $H_{ii}(\mathbf{x})$ ($i=1, \dots, n$) of the Hessian of the function $f(\mathbf{x})$. Next, the system removes from consideration any subbox \mathbf{X} for which a diagonal element $H_{ii}(\mathbf{X})$ of the
10 Hessian over the subbox \mathbf{X} is always negative, indicating that the function f is not convex over the subbox \mathbf{X} and consequently does not contain a global minimum within the subbox \mathbf{X} . In this variation, the system applies term consistency to each inequality $H_{ii}(\mathbf{x}) \geq 0$ ($i=1, \dots, n$) over the subbox \mathbf{X} .

In a further variation, the system applies box consistency to each
15 inequality $H_{ii}(\mathbf{x}) \geq 0$ ($i=1, \dots, n$) over the subbox \mathbf{X} .

In a variation on this embodiment, if the subbox \mathbf{X} is strictly feasible ($p_i(\mathbf{X}) < 0$ for all $i=1, \dots, n$), while performing the interval Newton step the system computes the Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{X})$ of the gradient of the function f evaluated as a function of a point \mathbf{x} over the subbox \mathbf{X} . Next, the system computes an
20 approximate inverse \mathbf{B} of the center of $\mathbf{J}(\mathbf{x}, \mathbf{X})$, and uses the approximate inverse \mathbf{B} to analytically determine the system $\mathbf{B}\mathbf{g}(\mathbf{x})$, wherein $\mathbf{g}(\mathbf{x})$ is the gradient of the function $f(\mathbf{x})$, and wherein $\mathbf{g}(\mathbf{x})$ includes components $g_i(\mathbf{x})$ ($i=1, \dots, n$).

In a further variation, the system applies term consistency to each component $(\mathbf{B}\mathbf{g}(\mathbf{x}))_i = 0$ ($i=1, \dots, n$) to solve for the variable x_i over the subbox \mathbf{X} .

25 In a further variation, the system applies box consistency to each component $(\mathbf{B}\mathbf{g}(\mathbf{x}))_i = 0$ ($i=1, \dots, n$) to solve for the variable x_i over the subbox \mathbf{X} .

In a variation on this embodiment, the system performs the interval Newton step on the John conditions.

In a variation on this embodiment, while performing the interval inequality constrained global optimization process, the system linearizes the set of inequality constraints to produce a set of linear inequality constraints with interval coefficients that enclose the nonlinear inequality constraints. Next, the system preconditions the set of linear inequality constraints through additive linear combinations to produce a set of preconditioned linear inequality constraints. In this variation, the system applies term consistency to the set of preconditioned linearized inequality constraints over the subbox X .

In a further variation, the system applies box consistency to the set of preconditioned linearized inequality constraints over the subbox X .

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 illustrates a computer system in accordance with an embodiment of the present invention.

FIG. 2 illustrates the process of compiling and using code for interval computations in accordance with an embodiment of the present invention.

FIG. 3 illustrates an arithmetic unit for interval computations in accordance with an embodiment of the present invention.

FIG. 4 is a flow chart illustrating the process of performing an interval computation in accordance with an embodiment of the present invention.

FIG. 5 illustrates four different interval operations in accordance with an embodiment of the present invention.

FIG. 6 is a flow chart illustrating the process of finding an interval solution to a nonlinear equation in accordance with an embodiment of the present invention.

FIG. 7A presents a portion of a flow chart illustrating the process of using term consistency to solve an interval global optimization problem with inequality constraints in accordance with an embodiment of the present invention.

5 FIG. 7B presents a portion of a flow chart illustrating the process of using term consistency to solve an interval global optimization problem with inequality constraints in accordance with an embodiment of the present invention.

FIG. 7C presents a portion of a flow chart illustrating the process of using term consistency to solve an interval global optimization problem with inequality constraints in accordance with an embodiment of the present invention.

10 FIG. 7D presents a portion of a flow chart illustrating the process of using term consistency to solve an interval global optimization problem with inequality constraints in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

15 The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications
20 without departing from the spirit and scope of the present invention. Thus, the present invention is not limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any device
25 or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs) and DVDs (digital versatile discs or

digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

5

Computer System

FIG. 1 illustrates a computer system 100 in accordance with an embodiment of the present invention. As illustrated in FIG. 1, computer system 100 includes processor 102, which is coupled to a memory 112 and a peripheral bus 110 through bridge 106. Bridge 106 can generally include any type of circuitry for coupling components of computer system 100 together.

Processor 102 can include any type of processor, including, but not limited to, a microprocessor, a mainframe computer, a digital signal processor, a personal organizer, a device controller and a computational engine within an appliance. Processor 102 includes an arithmetic unit 104, which is capable of performing computational operations using floating-point numbers.

Processor 102 communicates with storage device 108 through bridge 106 and peripheral bus 110. Storage device 108 can include any type of non-volatile storage device that can be coupled to a computer system. This includes, but is not limited to, magnetic, optical, and magneto-optical storage devices, as well as storage devices based on flash memory and/or battery-backed up memory.

Processor 102 communicates with memory 112 through bridge 106. Memory 112 can include any type of memory that can store code and data for execution by processor 102. As illustrated in FIG. 1, memory 112 contains computational code for intervals 114. Computational code 114 contains instructions for the interval operations to be performed on individual operands, or interval values 115, which are also stored within memory 112. This

computational code 114 and these interval values 115 are described in more detail below with reference to FIGs. 2-5.

Note that although the present invention is described in the context of computer system 100 illustrated in FIG. 1, the present invention can generally operate on any type of computing device that can perform computations involving floating-point numbers. Hence, the present invention is not limited to the computer system 100 illustrated in FIG. 1.

Compiling and Using Interval Code

FIG. 2 illustrates the process of compiling and using code for interval computations in accordance with an embodiment of the present invention. The system starts with source code 202, which specifies a number of computational operations involving intervals. Source code 202 passes through compiler 204, which converts source code 202 into executable code form 206 for interval computations. Processor 102 retrieves executable code 206 and uses it to control the operation of arithmetic unit 104.

Processor 102 also retrieves interval values 115 from memory 112 and passes these interval values 115 through arithmetic unit 104 to produce results 212. Results 212 can also include interval values.

Note that the term “compilation” as used in this specification is to be construed broadly to include pre-compilation and just-in-time compilation, as well as use of an interpreter that interprets instructions at run-time. Hence, the term “compiler” as used in the specification and the claims refers to pre-compilers, just-in-time compilers and interpreters.

Arithmetic Unit for Intervals

FIG. 3 illustrates arithmetic unit 104 for interval computations in more detail accordance with an embodiment of the present invention. Details regarding the construction of such an arithmetic unit are well known in the art. For example, see U.S. Patent Nos. 5,687,106 and 6,044,454. Arithmetic unit 104 receives intervals 302 and 312 as inputs and produces interval 322 as an output.

In the embodiment illustrated in FIG. 3, interval 302 includes a first floating-point number 304 representing a first endpoint of interval 302, and a second floating-point number 306 representing a second endpoint of interval 302.

Similarly, interval 312 includes a first floating-point number 314 representing a first endpoint of interval 312, and a second floating-point number 316 representing a second endpoint of interval 312. Also, the resulting interval 322 includes a first floating-point number 324 representing a first endpoint of interval 322, and a second floating-point number 326 representing a second endpoint of interval 322.

Note that arithmetic unit 104 includes circuitry for performing the interval operations that are outlined in FIG. 5. This circuitry enables the interval operations to be performed efficiently.

However, note that the present invention can also be applied to computing devices that do not include special-purpose hardware for performing interval operations. In such computing devices, compiler 204 converts interval operations into a executable code that can be executed using standard computational hardware that is not specially designed for interval operations.

FIG. 4 is a flow chart illustrating the process of performing an interval computation in accordance with an embodiment of the present invention. The system starts by receiving a representation of an interval, such as first floating-point number 304 and second floating-point number 306 (step 402). Next, the

system performs an arithmetic operation using the representation of the interval to produce a result (step 404). The possibilities for this arithmetic operation are described in more detail below with reference to FIG. 5.

5 Interval Operations

FIG. 5 illustrates four different interval operations in accordance with an embodiment of the present invention. These interval operations operate on the intervals X and Y . The interval X includes two endpoints,

\underline{x} denotes the lower bound of X , and
 \bar{x} denotes the upper bound of X .

10 The interval X is a closed subset of the extended (including $-\infty$ and $+\infty$) real numbers R^* (see line 1 of FIG. 5). Similarly the interval Y also has two endpoints and is a closed subset of the extended real numbers R^* (see line 2 of FIG. 5).

Note that an interval is a point or degenerate interval if $X = [x, x]$. Also note that the left endpoint of an interior interval is always less than or equal to the
15 right endpoint. The set of extended real numbers, R^* is the set of real numbers, R , extended with the two ideal points negative infinity and positive infinity:

$$R^* = R \cup \{-\infty\} \cup \{+\infty\}.$$

20 In the equations that appear in FIG. 5, the up arrows and down arrows indicate the direction of rounding in the next and subsequent operations. Directed rounding (up or down) is applied if the result of a floating-point operation is not machine-representable.

The addition operation $X + Y$ adds the left endpoint of X to the left
25 endpoint of Y and rounds down to the nearest floating-point number to produce a

resulting left endpoint, and adds the right endpoint of X to the right endpoint of Y and rounds up to the nearest floating-point number to produce a resulting right endpoint.

5 Similarly, the subtraction operation $X - Y$ subtracts the right endpoint of Y from the left endpoint of X and rounds down to produce a resulting left endpoint, and subtracts the left endpoint of Y from the right endpoint of X and rounds up to produce a resulting right endpoint.

10 The multiplication operation selects the minimum value of four different terms (rounded down) to produce the resulting left endpoint. These terms are: the left endpoint of X multiplied by the left endpoint of Y ; the left endpoint of X multiplied by the right endpoint of Y ; the right endpoint of X multiplied by the left endpoint of Y ; and the right endpoint of X multiplied by the right endpoint of Y . This multiplication operation additionally selects the maximum of the same four terms (rounded up) to produce the resulting right endpoint.

15 Similarly, the division operation selects the minimum of four different terms (rounded down) to produce the resulting left endpoint. These terms are: the left endpoint of X divided by the left endpoint of Y ; the left endpoint of X divided by the right endpoint of Y ; the right endpoint of X divided by the left endpoint of Y ; and the right endpoint of X divided by the right endpoint of Y . This division
20 operation additionally selects the maximum of the same four terms (rounded up) to produce the resulting right endpoint. For the special case where the interval Y includes zero, X/Y is an exterior interval that is nevertheless contained in the interval R^* .

25 Note that the result of any of these interval operations is the empty interval if either of the intervals, X or Y , are the empty interval. Also note, that in one embodiment of the present invention, extended interval operations never cause

undefined outcomes, which are referred to as “exceptions” in the IEEE 754 standard.

Term Consistency

5 FIG. 6 is a flow chart illustrating the process of solving a nonlinear equation through interval arithmetic and term consistency in accordance with an embodiment of the present invention. The system starts by receiving a representation of a nonlinear equation $f(\mathbf{x}) = 0$ (step 602), as well as a representation of an initial box with \mathbf{X} with X_j an element of \mathbf{X} (step 604). Next, 10 the system symbolically manipulates the equation $f(\mathbf{x}) = 0$ into a form $g(x_j) - h(\mathbf{x}) = 0$, wherein the term $g(x_j)$ can be analytically inverted to produce an inverse function $g^{-1}(y)$ (step 606).

Next, the system substitutes the initial box \mathbf{X} into $h(\mathbf{X})$ to produce the equation $g(X'_j) = h(\mathbf{X})$ (step 608). The system then solves for $X'_j = g^{-1}(h(\mathbf{X}))$ 15 (step 610). The resulting interval X'_j is then intersected with the initial interval X_j to produce a new interval X_j^+ (step 612).

At this point, if X_j^+ is empty, the system can terminate. Otherwise, the system can perform further processing. This further processing involves saving X_j by setting $X_j^{(0)} = X_j$ and also, setting $X_j = X_j^+$ (step 614). Next, if $w(X_j^{(0)})$ is 20 sufficiently reduced at step 616, the system returns to step 606 for another iteration of term consistency on another term g of $f(\mathbf{x})$. Otherwise, the system terminates.

Examples of Applying Term Consistency

25 For example, suppose $f(x) = x^2 - x + 6$. We can define $g(x) = x^2$ and $h(x) = x - 6$. Let $X = [-10, 10]$. The procedural step is $(X')^2 = X - 6 = [-16, 4]$. Since $(X')^2$ must be non-negative, we replace this interval by $[0, 4]$. Solving for X' , we

obtain $X' = \pm [0,2]$. In replacing the range of $h(x)$ (i.e., $[-16,4]$) by non-negative values, we have excluded that part of the range $h(x)$ that is not in the domain of $g(x) = x^2$.

Suppose that we reverse the roles of g and h and use the iterative step
5 $h(X') = g(X)$. That is $X' - 6 = X^2$. We obtain $X' = [6,106]$. Intersecting this result with the interval $[-10,10]$, of interest, we obtain $[6,10]$. This interval excludes the set of values for which the range of $g(X)$ is not in the intersection of the domain of $h(X)$ with X .

Combining these results, we conclude that any solution of
10 $f(X) = g(X) - h(X) = 0$ that occurs in $X = [-10,10]$ must be in both $[-2,2]$ and $[6,10]$. Since these intervals are disjoint, there can be no solution in $[-10,10]$.

In practice, if we already reduced the interval from $[-10,10]$ to $[-2,2]$ by solving for g , we use the narrower interval as input when solving for h .

This example illustrates the fact that it can be advantageous to solve a
15 given equation for more than one of its terms. The order in which terms are chosen affects the efficiency. Unfortunately, it is not known how best to choose the best order.

Also note that there can be many choices for $g(x)$. For example, suppose we use term consistency to narrow the interval bound X on a solution of
20 $f(x) = ax^4 + bx + c = 0$. We can let $g(x) = bx$ and compute $X' = -(aX^4 + c)/b$ or we can let $g(x) = ax^4$ and compute $X' = \pm [-(bX+c)/a]^{1/4}$. We can also separate x^4 into $x^2 * x^2$ and solve for one of the factors $X' = \pm [-(bX+c)/(aX^2)]^{1/2}$.

In the multidimensional case, we may solve for a term involving more than one variable. We then have a two-stage process. For example, suppose we solve
25 for the term $1/(x+y)$ from the function $f(x,y) = 1/(x+y) - h(x,y) = 0$. Let $x \in X = [1,2]$ and $y \in Y = [0.5,2]$. Suppose we find that $h(X,Y) = [0.5,1]$. Then $1/(x+y) \in [0.5,1]$ so $x+y \in [1,2]$. Now we replace y by $Y = [0.5,2]$ and obtain the

bound $[-1, 1.5]$ on X . Intersecting this interval with the given bound $X = [1, 2]$ on x , we obtain the new bound $X' = [1, 1.5]$.

We can use X' to get a new bound on h ; but this may require extensive computing if h is a complicated function; so suppose we do not. Suppose that we do, however, use this bound on our intermediate result $x + y = [1, 2]$. Solving for y as $[1, 2] - X'$, we obtain the bound $[-0.5, 1]$. Intersecting this interval with Y , we obtain the new bound $Y' = [0.5, 1]$ on y . Thus, we improve the bounds on both x and y by solving for a single term of f .

The point of these examples is to show that term consistency can be used in many ways both alone and in combination with the interval Newton algorithm to improve the efficiency with which roots of a single nonlinear equation can be computed. The same is true for systems of nonlinear equations.

Inequality Constrained Interval Global Optimization

FIGs. 7A-7D collectively present a flow chart illustrating the process of solving an interval global optimization problem with inequality constraints in accordance with an embodiment of the present invention. Generally, we seek a solution in a single box specified by the user. However, any number of boxes can be initially specified.

The boxes can be disjoint or overlap. However, if they overlap, a minimum at a point that is common to more than one box is separately found as a solution in each box containing it. In this case, computing effort is wasted. If the user does not specify an initial box or boxes, we use a default box. The process finds the global minimum in the set of points formed by the set of boxes. We assume these initial boxes are placed in a list L_1 of boxes to be processed.

Suppose the user of the process knows a point x_bar that is guaranteed to be feasible. If so, we use this point to compute an initial upper bound f_bar on the

global minimum f^* . If x_bar cannot be represented exactly on the computer, the system forms a representable interval vector \mathbf{X} containing x_bar . We evaluate $f(\mathbf{X})$ and obtain [lower bound $f(\mathbf{X})$, upper bound $f(\mathbf{X})$]. Even if rounding and/or dependence are such that \mathbf{X} cannot be numerically proven to be certainly feasible, we rely upon the user and assume that \mathbf{X} contains a feasible point. Therefore, we set f_bar equal to the upper bound of $f(\mathbf{X})$.

Also the user may know an upper bound f_bar on f^* even though he may not know where (or even if) f takes on such a value in the feasible region defined by the inequality constraints. If so, we set f_bar equal to this known bound. If the known bound is not representable on the computer, the system rounds the value up to a larger value that is representable.

If no feasible point is known and no upper bound on f^* is known, we set $f_bar = +\infty$. The user must specify a box size tolerance ϵ_X and a function width tolerance ϵ_F .

In the system, nonlinear functions are often linearized over a box \mathbf{X} using Taylor expansion. However, use of linearization is generally ineffective if \mathbf{X} is wide. Four different sub-procedures in the system use linearization. The system uses a "linearization test" to decide if a given sub-procedure should be used for a given box. Each of the four sub-procedures uses a separate test of the same kind. In each case, a criterion for "success" is defined. The symbol w_R denotes the width of the largest box for which success was achieved. The symbol w_I denotes the width of the smallest box for which success was not achieved. A given sub-procedure is applied for a box \mathbf{X} whenever $w(\mathbf{X}) \leq (w_R + w_I)/2$. For each sub-procedure, the system initially sets $w_R = 0$ and $w_I = w(\mathbf{X}^{(0)})$, where $\mathbf{X}^{(0)}$ is the initial box. In addition, the system specifies a bound f_bar if one is known. Note that the four sub-procedures referred to above are: (1) Newton applied to the

gradient; (2) Newton applied to the John conditions; (3) linearization of the constraints; and (4) linearization of $f(\mathbf{x}) \leq f_bar$.

The steps of the process are performed in the order given except as indicated by branching.

5 First, for each box in the list L_I , the system applies term consistency to each of the inequality constraints $p_i(\mathbf{x}) \leq 0$ ($i=1, \dots, m$) (step 701).

If $f_bar < +\infty$, then for each box in L_I , the system applies term consistency to the inequality $f \leq f_bar$ (step 702).

10 If L_I is empty, the system goes to step 742. Otherwise, the system selects (for the next box \mathbf{X} to be processed) the box in L_I for which the lower bound of $f(\mathbf{X})$ is smallest. For later reference, the system denotes this box by $\mathbf{X}^{(1)}$. The system also deletes \mathbf{X} from L_I (step 703).

The system applies term consistency over \mathbf{X} to each constraint inequality. If \mathbf{X} is deleted, the system goes to step 703. The system skips this step if \mathbf{X} has not changed since step 701. (step 704).

Next, the system computes an approximation \mathbf{x} for the center $m(\mathbf{X})$ of \mathbf{X} . If the upper bound of $f(\mathbf{x}) > f_bar$, the system goes to step 708 (step 705).

For future reference, the system denotes the box \mathbf{X} by $\mathbf{X}^{(2)}$. Next, the system does a constrained line search to try to reduce f_bar (step 706).

20 If f_bar was not reduced in step 706, the system goes to step 709 (step 707).

Next, the system applies term consistency to the inequality $f(\mathbf{x}) \leq f_bar$. If \mathbf{X} is deleted, the system goes to step 703 (step 708).

25 If $w(\mathbf{X}) < \epsilon_X$ and $w[f(\mathbf{X})] < \epsilon_F$, the system puts \mathbf{X} in list L_2 . Otherwise, if \mathbf{X} is sufficiently reduced relative to the box $\mathbf{X}^{(1)}$ defined in step 703, the system puts \mathbf{X} in L_I and goes to step 703 (step 709). We say that a box \mathbf{X} is sufficiently

reduced if any component of \mathbf{X} is reduced by an amount that is at least a fraction (say 0.25) of the width of the widest component of \mathbf{X} .

Next, the system applies box consistency to each inequality constraint. If $f_bar < +\infty$, the system also applies box consistency to the inequality $f(\mathbf{x}) \leq f_bar$.

5 If \mathbf{X} is deleted, the system goes to step 703 (step 710).

If the upper bound of $p_i(\mathbf{X}) \geq 0$ for any $i=1, \dots, n$, (i.e., if \mathbf{X} is not certainly strictly feasible), the system goes to step 726 (step 711).

Next, the system applies term consistency to $g_i(\mathbf{x})=0$ for $i=1, \dots, n$, where \mathbf{g} is the gradient of the objective function f . If the result for any $i=1, \dots, n$ is empty,
10 the system goes to step 703 (step 712). Note that the steps 712 through 725 do not use inequality constraints because none are active for the current box \mathbf{X} .

Otherwise, the system applies term consistency to the relation $H_{ii}(\mathbf{x}) \geq 0$ for $i=1, \dots, n$, where H_{ii} is a diagonal element of the Hessian of f . If the result is empty, the system goes to step 703 (step 713).

15 Next, the system repeats step 709 (step 714).

The system then applies box consistency to $g_i=0$ for $i=1, \dots, n$. If the result is empty, the system goes to step 703 (step 715).

Next, the system applies box consistency to $H_{ii}(\mathbf{x}) \geq 0$ for $i=1, \dots, n$. If the result is empty, the system goes to step 703 (step 716).

20 Next, the system repeats step 709 (step 717).

The system then uses a criterion $w(\mathbf{X}) > (w_I + w_R)/2$ to decide if a Newton step should be applied to solve $\mathbf{g}=0$. If not, the system goes to step 726 (step 718). Note that, w_I denotes the width of the smallest box for which $\mathbf{M}^I = \mathbf{BJ}(\mathbf{x}, \mathbf{X})$ is irregular. If \mathbf{M}^I is regular for a given box, w_R denotes the width
25 of the largest box for which \mathbf{M}^I has been shown to be regular.

The system generates the interval Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{X})$ of the gradient \mathbf{g} and computes the approximate inverse \mathbf{B} of the center of $\mathbf{J}(\mathbf{x}, \mathbf{X})$. The system also

applies one step of an interval Newton method to solve $\mathbf{g}=\mathbf{0}$. If the result is empty, the system goes to step 703 (step 719).

Next, the system repeats step 709 (step 720).

5 The system then uses the matrix \mathbf{B} found in step 719 to obtain \mathbf{Bg} in analytic form. The system applies term consistency to solve the i -th equation of $\mathbf{Bg}=0$ for the i -th variable x_i for $i=1,\dots,n$. If the result is empty, the system goes to step 703 (step 721).

Next, the system repeats step 709 (step 722).

10 The system uses box consistency to solve the i -th equation of \mathbf{Bg} (as obtained in step 721) for the i -th variable for $i=1,\dots,n$. If the result is empty, the system goes to step 703 (step 723).

Next, the system repeats step 709 (step 724).

15 The system uses the matrix \mathbf{B} found in step 719 in a Newton Step to try to reduce the upper bound f_bar (step 725). A line search can be performed as follows. Suppose we evaluate the gradient $\mathbf{g}(\mathbf{x})$ of $f(\mathbf{x})$ at a point \mathbf{x} . Note that f decreases (locally) in the negative gradient direction from \mathbf{x} . A simple procedure for finding a point where f is small is to search along this half-line. Let \mathbf{x} be the center of the current box. Define the half-line of points $\mathbf{y}(\alpha)=\mathbf{x}-\alpha\mathbf{g}(\mathbf{x})$ where $\alpha\geq 0$.

20 We now use a standard procedure for finding an approximate minimum of the objective function f on this half-line. We first restrict our region of search by determining the value α' such that $\mathbf{y}(\alpha')=\mathbf{x}-\alpha'\mathbf{g}$ is on the boundary of the current box \mathbf{X} , and search between \mathbf{x} and \mathbf{x}' . We use the following procedure. Each application of the procedure requires an evaluation of f . Procedure: If $f(\mathbf{x}')<f(\mathbf{x})$, replace \mathbf{x} by $(\mathbf{x}+\mathbf{x}')/2$. Otherwise, we replace \mathbf{x}' by $(\mathbf{x}+\mathbf{x}')/2$.

25 Next, the system computes an approximation \mathbf{x} for the center $m(\mathbf{X})$ of \mathbf{X} . If $f(\mathbf{x})>f_bar$, the system goes to step 703 (step 726).

The system skips this step and goes to step 732 if $\mathbf{X} = \mathbf{X}^{(2)}$, the same box for which a line search was done in step 706. Otherwise, the system does a line search to try to reduce f_bar . If f_bar is not reduced, the system goes to step 732 (step 727).

- 5 For future reference, the system denotes \mathbf{X} by $\mathbf{X}^{(3)}$. The system then uses a linearization test to decide whether to linearize and “solve” the inequality $f(\mathbf{x}) \leq f_bar$. If this condition is not satisfied, the system goes to step 732 (step 728).

- 10 The system uses a linear method to try to reduce \mathbf{X} using the inequality $f(\mathbf{x}) \leq f_bar$. If \mathbf{X} is deleted, the system goes to step 703. Otherwise, if this application of the linear method does not sufficiently reduce the box $\mathbf{X}^{(3)}$, the system goes to step 731 (step 729).

The system uses a quadratic method to try to reduce \mathbf{X} using the inequality $f(\mathbf{x}) \leq f_bar$. If \mathbf{X} is deleted, the system goes to step 703 (step 730).

- 15 Next, the system repeats step 709 (step 731).

The system uses a criterion similar to that in step 718 to decide whether to linearize and “solve” the inequality constraints. If the procedure indicates that the linearization should not be done, the system goes to 739 (step 732).

- 20 Next, the system selects the inequality constraints to be solved in linearized form, and possibly adds to this set the inequality $f(\mathbf{x}) \leq f_bar$. Note that the selection process removes from consideration any inequality constraints that are not sufficiently violated. If no inequalities are selected, the system goes to step 739. Otherwise, the system linearizes the resulting set of inequalities, and solves the resulting set of linear inequalities. If the solution set is empty, the
25 system goes to step 703 (step 733).

Next, the system repeats step 709 (step 734).

5 The system then uses the preconditioning matrix **B** formed at step 733 to analytically precondition the set of inequalities that were selected for use in step 733. The system also uses term consistency to solve each of the preconditioned inequalities. In so doing, each inequality is solved for the same (single) variable for which the linearized inequality was solved in step 733 (step 735).

Next, the system repeats step 709 (step 736).

The system uses box consistency to solve the same inequalities for the same variables as in step 735 (step 737).

Next, the system repeats step 709 (step 738).

10 The system uses a linearization test to decide whether to solve the John conditions. If not, the system goes to step 742 (step 739).

The system modifies the John conditions by omitting those constraints p_i for which $p_i(\mathbf{X}) < 0$ (since they are not active in \mathbf{X}). The system applies one pass of the interval Newton method to the (modified) John conditions. If the result is empty, the system goes to step 703 (step 740).

Next, the system repeats step 709 (step 741).

In various previous steps, gaps may have been generated in components of \mathbf{X} . If so, the system merges any of these gaps that overlap. The system then splits \mathbf{X} , and places the resulting subboxes in L_1 and goes to step 703 (step 742).

20 If $f_bar < +\infty$, the system applies term consistency to $f(\mathbf{x}) \leq f_bar$ for each box in the list L_2 . The system denotes those that remain by $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(s)}$ where s is the number of boxes remaining. The system also determines

$$\underline{F} = \min_{1 \leq i \leq s} \underline{f}(\mathbf{X}^{(i)}) \quad \text{and} \quad \overline{F} = \max_{1 \leq i \leq s} \overline{f}(\mathbf{X}^{(i)}) \quad (\text{step 743}).$$

25

Finally, the system terminates (step 744).

After termination, $w(\mathbf{X}) < \varepsilon_X$ and $w(f(\mathbf{X})) < \varepsilon_F$ for each remaining box \mathbf{X} in the list L_2 . Also,

$$\underline{F} \leq f(\mathbf{x}) \leq \overline{F}$$

for every point \mathbf{x} in all remaining boxes. If, after termination, $f_bar < +\infty$, we know there is a feasible point in the initial box(es). Therefore, we know that

$$\underline{F} \leq f^* \leq \min\{\overline{f}, \overline{F}\}.$$

If, after termination, $f_bar = +\infty$, then we have not found a certainly feasible point. There may or may not be one in $\mathbf{X}^{(0)}$. However, we know that if a feasible point \mathbf{x} does exist in $\mathbf{X}^{(0)}$, then

$$\underline{F} \leq f(\mathbf{x}) \leq \overline{F}.$$

Suppose a feasible point exists. If our algorithm fails to find a certainly feasible point, then it does not produce an upper bound f_bar and cannot use the relation $f \leq f_bar$. In particular, it cannot delete local minima where $f(\mathbf{x}) > f^*$. In this case, all local minima are contained in the set of output boxes.

If all of the initial box $\mathbf{X}^{(0)}$ is deleted by our process, then we have proved that every point in $\mathbf{X}^{(0)}$ is infeasible. Suppose that every point in $\mathbf{X}^{(0)}$ is infeasible. Our process may prove this to be the case. However, we delete a subbox of $\mathbf{X}^{(0)}$ only if it is *certainly* infeasible. Rounding errors and/or dependence may prevent us from proving certain infeasibility of an infeasible subbox. Increased wordlength can reduce rounding errors and decreasing ε_X can reduce the effect of dependence by causing subboxes to eventually become smaller. However, neither effect can completely be removed.

Suppose $f_bar = +\infty$ after termination and $\mathbf{X}^{(0)}$ has not been entirely eliminated. It may still be possible either to compute $f_bar < \infty$ or to delete all of $\mathbf{X}^{(0)}$ by reducing the values of ε_X and ε_F and continuing to apply the process. To try to do so, we need only to reduce these tolerances and move the boxes from list L_2 to list L_1 . We can then restart the algorithm from the beginning with or without use of increased precision.

Note that steps 712 through 725 are essentially the same as corresponding steps in the process for unconstrained optimization. This is because these steps are applied to a box that is certainly feasible.

In our process, we avoid using more complicated procedures until the simpler ones no longer make sufficient progress in reducing the current box. For example, we delay use of the John conditions until all other procedures become unproductive.

We avoid using procedures that use Taylor expansions until we have evidence that expanded forms provide sufficiently accurate approximations to functions.

Inequality constraints are often simple relations of the form $x_i \leq b_i$ or $x_i \geq a_i$. Such constraints serve to determine the initial box $\mathbf{X}^{(0)}$. Therefore, they are satisfied throughout $\mathbf{X}^{(0)}$. Such constraints are omitted when applying any procedure designed to eliminate infeasible points. See steps 701, 704, 710 and 733.

In step 706 we use a line search to try to reduce f_bar . This involves evaluating the gradient of f . We can avoid this evaluation by simply checking if the midpoint \mathbf{x} of the box is feasible and, if so, using $f(\mathbf{x})$ as a candidate value for f_bar . However, it helps to have a finite value of f_bar early, so the line search is worth doing when $f_bar = +\infty$. Step 727 also uses a line search. It is less important here because a finite value of f_bar is likely to be computed in step 706. If there

are a large number of constraints, then evaluating the gradient is not a dominant part of the work to do the line search.

Experience has shown that efficiency is enhanced if the subbox \mathbf{X} to be processed is chosen to be the one for which $\inf(f(\mathbf{X}))$ is smallest among all candidate subboxes. This tends to cause a smaller value of f_bar to be computed early in the algorithm. Therefore, we return to step 703 to choose a new subbox whenever the current box has substantially changed.

Suppose we find that $p_i(\mathbf{X}) \leq 0$ for some value of i and some box \mathbf{X} . Then $p_i(\mathbf{X}') \leq 0$ for any $\mathbf{X}' \subseteq \mathbf{X}$. Therefore, we record the fact that $p_i(\mathbf{X}) \leq 0$ so that we need not evaluate $p_i(\mathbf{X}')$ for any $\mathbf{X}' \subseteq \mathbf{X}$.

It is possible that the procedures in step 719, 721, 723 or 740 prove the existence of a solution to the optimization problem. If so, the user can be informed of this fact. Such a solution may be local or global.

The foregoing descriptions of embodiments of the present invention have been presented only for purposes of illustration and description. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. For example, although the present invention describes the use of derivatives in certain situations, it is often possible to use slopes instead of derivatives.

Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.